# AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

## Listing of Claims:

1      1. (Currently amended) A method for generating code to perform

2 anticipatory prefetching for data references, comprising:

3      receiving code to be executed on a computer system;

4      analyzing the code to identify data references to be prefetched, wherein the

5 data references are identified from basic blocks within *if* conditions regardless of

6 whether the basic blocks are likely to execute, and wherein analyzing the code

7 involves,

8      performing a first marking phase in which only data

9      references located in blocks that are certain to execute are

10      considered in determining which data references are covered by

11      preceding data references, and

12      performing a second marking phase in which data

13      references that are located in blocks that are not certain to execute

14      are considered; and

15      inserting prefetch instructions into the code in advance of the identified

16 data references, wherein inserting prefetch instructions includes inserting multiple

17 redundant prefetch instructions for a given data reference;

18      wherein inserting multiple redundant prefetch instructions involves

19 inserting the multiple redundant prefetch instructions into unused instruction slots,

20 and wherein executing multiple redundant prefetch instructions potentially avoids

21 a cache miss.

1  2. (Original) The method of claim 1, further comprising:

2    profiling execution of the code to produce profiling results; and

3    using the profiling results to determine whether a given block of

4 instructions is executed frequently enough to perform the second marking phase

5 on the given block of instructions.


1  3. (Original) The method of claim 2, wherein determining whether the

2 given block of instructions is executed frequently enough to perform the second

3 marking phase involves comparing a frequency of execution for the given block

4 from the profiling results with a threshold value indicating a minimum frequency

5 of execution to be considered in the second marking phase.


1  4. (Original) The method of claim 1, wherein analyzing the code involves:

2    identifying loop bodies within the code; and

3    identifying data references to be prefetched from within the loop bodies.


1  5. (Original) The method of claim 4, wherein if there exists a nested loop

2 within the code, analyzing the code involves:

3    examining an innermost loop in the nested loop; and

4    examining a loop outside the innermost loop if the innermost loop is

5 smaller than a minimum size or is executed fewer than a minimum number of

6 iterations.


1  6. (Original) The method of claim 4, wherein analyzing the code to

2 identify data references to be prefetched involves examining a pattern of data

3 references over multiple loop iterations.


3

1    7. (Original) The method of claim 1, wherein analyzing the code involves
2    analyzing the code within a compiler.


1    8. (Currently amended) A computer-readable storage medium storing
2    instructions that when executed by a computer cause the computer to perform a
3    method for generating code to perform anticipatory prefetching for data
4    references, the method comprising:
5        receiving code to be executed on a computer system;
6        analyzing the code to identify data references to be prefetched, _wherein the
7    data references are identified from basic blocks within _if_ conditions regardless of
8    whether the basic blocks are likely to execute, and_ wherein analyzing the code
9    involves,
10                performing a first marking phase in which only data
11               references located in blocks that are certain to execute are
12               considered in determining which data references are covered by
13               preceding data references, and
14               performing a second marking phase in which data
15               references that are located in blocks that are not certain to execute
16               are considered; and
17       inserting prefetch instructions into the code in advance of the identified
18    data references, wherein inserting prefetch instructions includes inserting multiple
19    redundant prefetch instructions for a given data reference;
20       wherein inserting multiple redundant prefetch instructions involves inserting the
21    multiple redundant prefetch instructions into unused instruction slots, and wherein
22    executing multiple redundant prefetch instructions potentially avoids a cache miss.


1    9. (Original) The computer-readable storage medium of claim 8, wherein
2    the method further comprises:

4

3    profiling execution of the code to produce profiling results; and

4    using the profiling results to determine whether a given block of

5 instructions is executed frequently enough to perform the second marking phase

6 on the given block of instructions.


1    10. (Original) The computer-readable storage medium of claim 9, wherein

2 determining whether the given block of instructions is executed frequently enough

3 to perform the second marking phase involves comparing a frequency of

4 execution for the given block from the profiling results with a threshold value

5 indicating a minimum frequency of execution to be considered in the second

6 marking phase.


1    11. (Original) The computer-readable storage medium of claim 8, wherein

2 analyzing the code involves:

3    identifying loop bodies within the code; and

4

5    dentifying data references to be prefetched from within the loop bodies.


1    12. (Original) The computer-readable storage medium of claim 11,

2 wherein if there exists a nested loop within the code, analyzing the code involves:

3    examining an innermost loop in the nested loop; and

4    examining a loop outside the innermost loop if the innermost loop is

5 smaller than a minimum size or is executed fewer than a minimum number of

6 iterations.


1    13. (Original) The computer-readable storage medium of claim 11,

2 wherein analyzing the code to identify data references to be prefetched involves

3 examining a pattern of data references over multiple loop iterations.

5

1      14. (Original) The computer-readable storage medium of claim 11,

2      wherein analyzing the code involves analyzing the code within a compiler.


1      15. (Currently amended) An apparatus that generates code to perform

2      anticipatory prefetching for data references, comprising:

3      a receiving mechanism that is configured to receive code to be executed on

4      a computer system;

5      an analysis mechanism that is configured to analyze the code to identify

6      data references to be prefetched, wherein the data references are identified from

7      basic blocks within *if* conditions regardless of whether the basic blocks are likely

8      to execute, and wherein the analysis mechanism is configured to,

9      perform a first marking phase in which only data references

10      located in blocks that are certain to execute are considered in

11      determining which data references are covered by preceding data

12      references, and to

13      perform a second marking phase in which data references

14      that are located in blocks that are not certain to execute are

15      considered; and

16      an insertion mechanism that is configured to insert prefetch instructions

17      into the code in advance of the identified data references, wherein inserting

18      prefetch instructions includes inserting multiple redundant prefetch instructions

19      for a given data reference;

20      wherein inserting multiple redundant prefetch instructions involves

21      inserting the multiple redundant prefetch instructions into unused instruction slots,

22      and wherein executing multiple redundant prefetch instructions potentially avoids

23      a cache miss.

1        16. (Original) The apparatus of claim 15, further comprising a profiling

2    mechanism that is configured to profile execution of the code to produce profiling

3    results;

4        wherein the analysis mechanism is configured to use the profiling results

5    to determine whether a given block of instructions is executed frequently enough

6    to perform the second marking phase on the given block of instructions.


1        17. (Original) The apparatus of claim 16, wherein the analysis mechanism

2    is configured to compare a frequency of execution for the given block from the

3    profiling results with a threshold value indicating a minimum frequency of

4    execution to be considered in the second marking phase.


1        18. (Original) The apparatus of claim 15, wherein the analysis mechanism

2    is configured to:

3        identify loop bodies within the code; and to

4        identify data references to be prefetched from within the loop bodies.


1        19. (Original) The apparatus of claim 18, wherein if there exists a nested

2    loop within the code, the analysis mechanism is configured to:

3        examine an innermost loop in the nested loop; and to

4        examine a loop outside the innermost loop if the innermost loop is smaller

5    than a minimum size or is executed fewer than a minimum number of iterations.


1        20. (Original) The apparatus of claim 18, wherein the analysis mechanism

2    is configured to examine a pattern of data references over multiple loop iterations.


1        21. (Original) The apparatus of claim 15, wherein the apparatus resides

2    within a compiler.

7

1          22-45 (Canceled).